UNITED STATES UTILITY PATENT APPLICATION

FOR

A Method, System, and Apparatus for Explicit Control over a Disk Cache Memory

Inventors:

Sanjeev N. Trika
Robert J. Royer, Jr.

Prepared by:
Michael Nesheiwat
Patent Attorney

Intel Corporation
2111 N.E. 25th Avenue; JF3-147
Hillsboro, OR 97124
Phone: (503) 712-8918
Facsimile: (503) 264-1729

Express Mail No. EV 325 527 706 US

BACKGROUND

1. Field

The present disclosure pertains to the field of disk cache memory. More particularly, the present disclosure pertains to memory management control over a disk cache memory.

[0001] 2. Description of Related Art

[0002] Various applications, such as, databases, computer games and three dimensional (3D) world navigations, require large low-latency memory storage requirements. The preceding applications include complex data structures in order to best utilize the limited available system memory. Typically, it is advantageous for applications to store this data in non-volatile memory since the data is preserved despite system crashes, reboots, or a power fail condition.

[0003] Typically, disk drives have been utilized for the preceding applications. However, disk drives have very high latency (wait time for memory operation to be completed). Consequently, the high latency results in poor performance due to constantly retrieving data from the disk drive. Another solution is to use system main memory. However, system main memory is expensive and volatile. Therefore, this will result in low storage capacity and a short data lifetime. Another solution is to use disk caches with a non-volatile type of cache memory. However, cache management policies tend to be inefficient due to pre-configured caching policies and suffer from lack of control over the disk cache memory.

Brief Description of the Figures

[0004] The present invention is illustrated by way of example and not limitation in the Figures of the accompanying drawings.

[0005] Figure 1 illustrates an apparatus utilized in accordance with an embodiment

[0006] Figure 2 illustrates a software diagram utilized in accordance with an embodiment.

[0007] Figure 3 illustrates a flowchart for a method in accordance with one embodiment.

Detailed Description

[0008] The following description provides method, system and apparatus for a cache to be utilized for providing applications with explicit cache memory management control. In the following description, numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate logic circuits, data structures, and software algorithms without undue experimentation.

[0009] As previously described, various problems exist for applications that require large non-volatile storage requirements with low latency. In contrast, in one aspect, the claimed subject matter utilizes a non-volatile cache memory (NV Cache) in between a main memory and a mass storage, such as, a disk drive in one embodiment. Also, the claimed subject matter depicts a software organization that enables applications for accessing an interface that is exposed by the NV cache driver for reserving a portion of the NV cache for application use. Therefore, the claimed subject matter also facilitates and provides the application to have explicit control over the data in the NV cache. In one embodiment, the NV cache is controlled by a plurality of predetermined functions that are supported by a driver for indirect or direct application calls. The predetermined functions are in addition to a standard I/O driver interface that is utilized by an operating system (OS), and can follow any standard memory management model. For example, some of the predetermined functions are:

- Allocate (), Get (), Set(), and Free() function calls

- File creation/deletion and read/write operations.

However, the claimed subject matter is not limited to the function names depicted. Rather, one skilled in the art appreciates utilizing different names for the predetermined functions that serve the same purpose. For example, an Allocate function that serves the purpose of allocating a portion of the memory may be called by a different name, such as, Reserve function that serves the same purpose. Likewise, a Get function that serves the purpose of reading a portion of the memory may be called by a different name, such as, a Read function that serves the same purpose.

[0010]

Figure 1 illustrates an apparatus utilized in accordance with an embodiment. In one aspect and embodiment, the apparatus depicts a novel architecture that enables a non-volatile cache (NV cache) to be coupled in between a main memory and a main storage with applications controlling the cache policy and/or a portion of the cache for explicit control by the application. In one embodiment, the apparatus may be implemented in a memory controller. In another embodiment, the apparatus may be implemented in a chipset. In yet another embodiment, the apparatus may be implemented in an application specific integrated circuit (ASIC). Also, the apparatus may be controlled or supervised by a driver in a software algorithm. For example, the software may be stored in an electronically-accessible medium that includes any mechanism that provides (i.e., stores and/or transmits) content (e.g., computer executable instructions) in a form readable by an electronic device (e.g., a computer, a personal digital assistant, a cellular telephone).For example, a machine-accessible medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash

memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals).

[0011] The main memory 104 is coupled to a central processor unit (CPU) 102 and receives requests to access data. In some instances, the main memory will not have the requested data and forwards the request to the data store. In one embodiment, the NV cache 106 is coupled in between the main memory and the data store to receive the request. Also, in the same embodiment, the mass storage is a disk drive.

[0012] In one embodiment, the NV cache is manufactured with a polymer memory.

[0013] In order to illustrate the operation of this apparatus, the next few figures and examples will clearly illustrate the operation of the NV cache.

[0014] Figure 2 illustrates a software diagram utilized in accordance with an embodiment. In one aspect, the software diagram depicts a software organization that enables applications for directly accessing an interface that is exposed by the NV cache drive for reserving a portion of the NV cache for application use. For example, the software organization depicts a mechanism and interface to an NV cache (as depicted earlier in connection with Figure 1) for application use. Therefore, the software organization provides applications with explicit management control of the disk cache memory, NV cache.

For example, the software may be stored in an electronically-accessible medium includes any mechanism that provides (i.e., stores and/or transmits) content (e.g., computer executable instructions) in a form readable by an electronic device (e.g., a computer, a personal digital assistant, a cellular telephone).For example, a machine-accessible medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage

media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals).

[0015] As previously discussed, the claimed subject matter describes various functions (interfaces) which the NVcache will implement. One such set of function may comprise the following:

- Allocate (), Get (), Set(), and Free() function calls
- File creation/deletion and read/write operations

The allocate function (may also be described as an "interface") facilitates reserving a predetermined portion of the NVcache for an explicit control by the application. For example, the allocate function may be used as follows:

```
Int numBytesToReserve = 4000; (allocating # of bytes)
NvCacheReservedMem_t* pNvMem;
Byte arrBytes [4000];

pNvMem = Allocate (numBytesToReserve);
if (pNvMem is not NULL) { if allocation was successful)
     then use the reserved(allocated) memory
```

The preceding code illustrates one example of allocating 4000 bytes of memory (based on the variable numBytesToReserve). However, the claimed subject matter is not limited to neither allocating 4000 bytes or to the specific function, variable or date structure names. One skilled in the art appreciates the ease and simplicity of choosing another number of bytes with a different value for the variable numBytesToReserve based at least in part on the application type. In one embodiment, the claimed subject matter determines whether the allocation was successful by checking the status of pNvMem. If the allocation was successful, the set function is initiated. For example, the set function may be used as follows:

initialize arrBytes to data you want to store in pNvMem.
Set (pNvMem, offset=0, numBytes=4000, arrBytes); (using the memory by initializing to the values stored in arrBytes)

The set function allows one to initialize (write) the allocated number of bytes to a predetermined value. For example, the predetermined value may be stored in arrBytes. Likewise, one may initialize either a subset of allocated bytes or the entire set of allocated bytes. Consequently, the set function results in initializing a subset or entire set of reserved non-volatile memory to predetermined values. Subsequently, the Get function allows one to read the value of the allocated bytes. The Get function could be utilized after the Set function or at a later point in time. Also, the Set and Get functions may be utilized once or multiple times on the same or different portions of the non-volatile cache for the same and/or different applications. For example, the Get function may be used as follows:

Get (pNvMem, offset=0, numBytes=4000, arrBytes);(Doing the read from memory)

... and at some other time you want to release this memory from the nvCace:

Upon completion of a particular application or in the event one decides to utilize the allocated bytes for another application or the application does not want to exist for the next iteration, the allocated bytes may be released (deallocated) by utilizing the following Free function:

Free (pNvMem).

As previously discussed in Figure 1, the NV cache 106 may support the predetermined functions of Allocate, Get, Set, and Free by performing the following actions. In one embodiment, a pin bit is stored per cache-line in the cache-line metadata and the data allocation is done on a cache-line granularity.

For the allocate, the cache will:

Identify if we can reserve so many bytes. If not, return NULL.
Identify cache-lines to use to reserve these bytes.
Flush these cache-lines if they are dirty.
Mark them empty.
Pin these cache-lines
Return a pointer to a structure that identifies the cache-lines reserved for this request.

For the set function the cache will:

NvCacheSet (pNvMem, offset, numBytes, dataBuffer)
Ensure that input params are valid (i.e., not null, and the data region referenced is in range)
Identify the cache-lines to use.
Copy data from dataBuffer to the applicable cache lines
Mark these lines valid (not empty).

For the Get function the cache will:

NvCacheGet (pNvMem, offset, numBytes, dataBuffer)
Ensure that input params are valid (i.e., not null, and the data region referenced is in range)
Identify the cache-lines to use, and ensure they are valid (not empty).
Copy data from the applicable cache lines into dataBuffer

For the Free function the cache will:

NvCacheFree (pNvMem)
Validate input param
Unpin the cache-lines
Mark them invalid

In an alternate embodiment, initially reserving a section of the cache memory upfront to be dedicated for application requests. The cache will then perform the following tasks to facilitate the function:

Runs a memory manager on the reserved section of the cache memory to satisfy the application requests. The cache may also dynamically change the size of the reserved

portion of the cache memory for best utilization of the cache.

[0016]

[0017] Various embodiments of the claimed subject matter may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions, which may be used to program a computer, or other electronic devices, to perform processes according to various embodiments. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, magneto-optical disks, Read-only memory (ROMs), Random-only memory (RAM), Erasable Programmable only memory (EPROMs), Electrically Erasable Programmable only memory (EEPROMs), magnetic or optical cards, flash memory, or another type of media/machine-readable medium suitable for storing electronic instructions.

[0018] The software organization 200 depicts an application 202, such as, but not limited to, a database, three dimensional (3D) Navigator, or video game. The application communicates with an operating system file system 204 and an NV cache Driver 206 directly or indirectly through a software library to perform a function. In one embodiment, the NV cache driver resides in a kernel space. In one embodiment, the application generates an message to the NV cache driver to perform a function. For example, one function may be that the NV cache driver could reserve a first portion of the NV cache 208 to be exclusively used for memory requests for the particular application, while a second portion of the NV cache is utilized as a disk cache and is coupled to the disk drive.

[0019] Figure 3 illustrates a flowchart for a method in accordance with one embodiment. In one aspect, the flowchart depicts reserving a portion of a cache for application memory requests.

A first portion of the cache is reserved for application memory requests, as illustrated by a block 302. The reserving of a portion of the cache may be used once or repeated multiple times for the same or different portions of the non-volatile cache for the same and/or different applications. In contrast, in one embodiment, a second portion of the cache is reserved to be used as a disk cache, as illustrated by a block 304. In this embodiment, the second portion of the cache is coupled to a disk drive. In an alternative embodiment, the second portion of the cache is reserved for another application rather than for a disk cache. Also, in one embodiment, the application may be a database, 3D navigator, or game. However, the claimed subject matter is not limited to the previous applications.

[0020]

While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.